

BBBBBBBBBBBB		000000000		000000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		000000000		000000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		000000000		000000000		TTTTTTTTTTTT		SSSSSSSSSS
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000000000		000000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		000000000		000000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		000000000		000000000		TTT	SSSSSSSSSS	

```
BBBBBBBBB 000000 000000 TTTTTTTTTT IIIIII 000000
BBBBBBBBB 000000 000000 TTTTTTTTTT IIIIII 000000
BB      BB 00      00 00      00  TT      II 00      00
BB      BB 00      00 00      00  TT      II 00      00
BB      BB 00      00 00      00  TT      II 00      00
BB      BB 00      00 00      00  TT      II 00      00
BBBBBBBBB 00      00 00      00  TT      II 00      00
BBBBBBBBB 00      00 00      00  TT      II 00      00
BB      BB 00      00 00      00  TT      II 00      00
BB      BB 00      00 00      00  TT      II 00      00
BB      BB 00      00 00      00  TT      II 00      00
BB      BB 00      00 00      00  TT      II 00      00
BBBBBBBBB 000000 000000 TTT      IIIIII 000000
BBBBBBBBB 000000 000000 TTT      IIIIII 000000
          ...
          ...
          ...
```

```
LL          IIIIII SSSSSSSS
LL          IIIIII SSSSSSSS
LL          II     SS
LL          II     SS
LL          II     SS
LL          II     SS
LL          II     SSSSSS
LL          II     SSSSSS
LL          II     SS
LL          II     SS
LL          II     SS
LL          II     SS
LLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLL IIIIII SSSSSSSS
```


BOOTIO
Table of contents

- BOOTSTRAP FILEREAD IO MODULE

L 5

15-SEP-1984 23:41:42 VAX/VMS Macro V04-00

Page 0

(1)	66	RDWRTLBN - READ/WRITE LOGICAL BLOCK NUMBER
(1)	130	BOO\$CACHE_INIT - INIT FILEREAD CACHE
(1)	264	BOO\$IMAGE_ATT - Get image attributes from image header
(1)	310	SYSS\$ASSIGN, Dummy assign device system service
(1)	347	Common Globals for VMB and SYSBOOT

```

0000 1      .TITLE  BOOTIO - BOOTSTRAP FILEREAD IO MODULE
0000 2      .IDENT  'V04-000'
0000 3      :
0000 4      :*****
0000 5      :
0000 6      :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7      :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8      :*  ALL RIGHTS RESERVED.
0000 9      :
0000 10     :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11     :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12     :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13     :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14     :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15     :*  TRANSFERRED.
0000 16     :
0000 17     :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18     :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19     :*  CORPORATION.
0000 20     :
0000 21     :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22     :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23     :
0000 24     :
0000 25     :*****
0000 26     :
0000 27     :++
0000 28     : FACILITY:      SYSTEM BOOTSTRAPPING
0000 29     :
0000 30     : ABSTRACT:
0000 31     :
0000 32     :      THIS MODULE PERFORMS LOGICAL BLOCK I/O FOR FILEREAD
0000 33     :
0000 34     : ENVIRONMENT:  KERNEL MODE, UNMAPPED, IPL=31
0000 35     :
0000 36     : AUTHOR:  RICHARD I. HUSTVEDT   , CREATION DATE: 14-APR-78
0000 37     :
0000 38     : MODIFIED BY:
0000 39     :
0000 40     :      V03-002 KDM0097      Kathleen D. Morse      09-Apr-1984
0000 41     :      Bias the out of range value by the initial page number.
0000 42     :
0000 43     :--

```



```

0000 45 :
0000 46 : INCLUDE FILES:
0000 47 :
0000 48 : $IHDEF ; IMAGE HEADER DEFINITIONS
0000 49 : $IHSDEF ; IMAGE HEADER SYMBOL TABLE DEFS
0000 50 : $IHPDEF ; IMAGE HEADER PATCH CONTROL DEFS
0000 51 : $RPBDEF ; DEFINE RESTART PARAMETER BLOCK
0000 52 :
0000 53 : MACROS:
0000 54 :
0000 55 : Define Memory Size to File Cache Parameter table entry
0000 56 :
0000 57 : .MACRO MEM_FILE_CACHE MEM_PAGE_CNT,CACHE_PAGE_NUM,CACHE_PAGE_CNT,MAX_PAGE
0000 58 : .LONG MEM_PAGE_CNT-<MEM_PAGE_CNT/10>
0000 59 : .WORD CACHE_PAGE_NUM
0000 60 : .WORD <<CACHE_PAGE_CNT+3>8^C<3>>
0000 61 : .LONG MAX_PAGE
0000 62 : .ENDM MEM_FILE_CACHE

```

```

00000000 64      .PSECT  YFILEREAD,BYTE,EXE
0000      65
0000      66      .SBTTL  RDWRTLBN - READ/WRITE LOGICAL BLOCK NUMBER
0000      67  :++
0000      68  : FUNCTIONAL DESCRIPTION:
0000      69  :
0000      70  :     THIS ROUTINE READS/Writes N BYTES FROM/TO THE SPECIFIED
0000      71  : LOGICAL BLOCK NUMBER OF THE VOLUME ASSIGNED TO THE SPECIFIED CHANNEL
0000      72  :
0000      73  : CALLING SEQUENCE:
0000      74  :
0000      75  :     CALLG  ARGLIST,FIL$RDWRTLBN
0000      76  :
0000      77  : INPUT PARAMETERS:
0000      78  :
0000      79  :     CHAN(AP)      =           ;CHANNEL ASSIGNED TO THE VOLUME TO READ
0000      80  :     LBN(AP)      =           ;LOGICAL BLOCK NUMBER TO READ
0000      81  :     BUFADR(AP)   =           ;ADDRESS OF BUFFER TO READ INTO
0000      82  :     IOFUNC(AP)  =           ;I/O FUNCTION CODE
0000      83  :     BYTCNT(AP)  =           ;NUMBER OF BYTES TO TRANSFER
0000      84  :
0000      85  : IMPLICIT INPUTS:
0000      86  :
0000      87  :     NONE
0000      88  :
0000      89  : OUTPUT PARAMETERS:
0000      90  :
0000      91  :     R0 = SYSTEM STATUS CODE
0000      92  :
0000      93  : IMPLICIT OUTPUTS:
0000      94  :
0000      95  :     NONE
0000      96  :
0000      97  : COMPLETION CODES:
0000      98  :
0000      99  :     NONE
0000     100  :
0000     101  : SIDE EFFECTS:
0000     102  :
0000     103  :     NONE
0000     104  :
0000     105  : EQUATED SYMBOLS:
0000     106  :
0000     107  :     OFFSETS FROM AP
0000     108  :
00000004 0000     109  :     CHAN      =           4           ;CHANNEL TO WHICH VOLUME IS ASSIGNED
00000008 0000     110  :     LBN      =           8           ;LOGICAL BLOCK NUMBER
0000000C 0000     111  :     BUFADR   =          12          ;BUFFER ADDRESS TO READ INTO
00000010 0000     112  :     IOFUNC   =          16          ;FUNCTION CODE FOR THE QIO
00000014 0000     113  :     BYTCNT   =          20          ;NUMBER OF BYTES TO TRANSFER
0000     114  :
0000     115  : --
0000     116  :
0000     117  : FIL$RDWRTLBN::
0000     118  :     .WORD    0
0000     119  :     PUSHL   CHAN(AP)
0000     120  :     MOVL    (SP),R0
0000     120  :     : ADDRESS OF RPB
0000     120  :     : GET ADDRESS OF RPB
04 AC DD 0002
50 6E DO 0005

```


BOOTIO
V04-000

- BOOTSTRAP FILEREAD IO MODULE C 6
RDWRTLBN - READ/WRITE LOGICAL BLOCK NUMB
15-SEP-1984 23:41:42 VAX/VMS Macro V04-00
4-SEP-1984 23:02:54 [BOOTS.SRC]BOOTIO.MAR;1

Page 4
(1)

50	34	A0	D0	0008	121	MOVL	RPBSL_IOVEC(R0),R0	:	GET POINTER TO I/O ROUTINE VECTOR
		00	DD	000C	122	PUSHL	#0	:	SET MODE TO PHYSICAL ADDRESS
	10	AC	DD	000E	123	PUSHL	IOFUNC(AP)	:	SET FUNCTION
	08	AC	DD	0011	124	PUSHL	LBN(AP)	:	LOGICAL BLOCK NUMBER
	14	AC	DD	0014	125	PUSHL	BYTCNT(AP)	:	SET NUMBER OF BYTES
	0C	BC	DF	0017	126	PUSHAL	@BUFADR(AP)	:	SET BUFFER ADDRESS
00	B040	06	FB	001A	127	CALLS	#6,@(R0)[R0]	:	CALL BOOTSTRAP DRIVER
			04	001F	128	RET			

```

0020 130 .SBTTL BOO$CACHE_INIT - INIT FILEREAD CACHE
0020 131 :++
0020 132 :
0020 133 : Functional description:
0020 134 :
0020 135 :     This routine establishes a desired FILEREAD cache size and
0020 136 :     base address according to the size of memory. It finds
0020 137 :     good contiguous pages at or near the desired place and
0020 138 :     calls the FIL$CACHE_INIT routine to initialize the cache.
0020 139 :     The routine is further divided into two pieces: one to do
0020 140 :     cache allocation, and one to do the actual mount and open.
0020 141 :     This is necessary for VMB needs to allocate the cache long
0020 142 :     before it is ready to accept IO to the device.
0020 143 :
0020 144 : Calling Sequence:
0020 145 :
0020 146 :     BSBW    BOO$CACHE_INIT
0020 147 :
0020 148 : Inputs:
0020 149 :
0020 150 :     R11                - RPB base address
0020 151 :     RPB$$_PFNCNT(R11)  - actual number of good pages in memory
0020 152 :     RPB$$_PFNMAP+4(R11) - base address of PFN bitmap
0020 153 :
0020 154 : Implicit inputs:
0020 155 :
0020 156 :     none
0020 157 :
0020 158 : Outputs:
0020 159 :
0020 160 :     R0-R4 altered
0020 161 :     FIL$GQ_CACHE set up with size and address of cache
0020 162 :
0020 163 : Implicit outputs:
0020 164 :
0020 165 : --
0020 166 :
0020 167 : Table of memory sizes to file cache parameters
0020 168 :
0020 169 : NOTE: If this table is modified, a corresponding table in VMB around
0020 170 : label MEM_TAB should be checked for consistency.
0020 171 :
0020 172 MEM_CACHE TABLE:
0020 173 MEM_FILE_CACHE 16384,2048,64,4096 ; More than 8 megabyte
0020 174 MEM_FILE_CACHE 8192,1024,64,2048 ; More than 4 megabyte
0020 175 MEM_FILE_CACHE 4096, 640,64,1024 ; More than 2 megabyte
0020 176 MEM_FILE_CACHE 2048, 512,64, 768 ; More than 1 megabyte
0020 177 MEM_FILE_CACHE 1024, 256,32, 512 ; More than 512k bytes
0020 178 MEM_FILE_CACHE 512, 256,16, 256 ; More than 256k bytes
0020 179 MEM_FILE_CACHE 384, 256, 8, 192 ; More than 192k bytes
0020 180 MEM_FILE_CACHE 256, 128, 4, 128 ; More than 128k bytes
0020 181 MEM_FILE_CACHE 0, 0, 0, 0 ;

```



```

008C 183 :
008C 184 : BOO$CACHE_ALLOC - The piece that does the allocation.
008C 185 :
008C 186 : Outputs:
008C 187 : FIL$GQ_CACHE filled in with size/address in blocks
008C 188 :
008C 189 BOO$CACHE_ALLOC::
50 0000'CF DD 008C 190 PUSHL R5 ; Save a register
51 8B AF 7C 008E 191 CLRQ W^FIL$GQ_CACHE ; Assume no cache available
51 80 7D 0092 192 MOVAL B^MEM_CACHE_TABLE,R0 ; Adr of memory size to cache params tbl
51 1F 13 0096 193 10$: MOVQ (R0)+,R1 ; Get the next table entry
51 80 D0 0099 194 BEQL 20$ ; Branch if memory too small for cache
51 4C AB D1 009B 195 MOVL (R0)+,R5 ; Max page
51 52 F2 19 009E 196 CMLP RPBSL_PFNcnt(R11),R1 ; More memory than this entry?
51 50 3C 00A2 197 BLSS 10$ ; Branch if not, get next one
51 52 FO 8F 78 00A7 198 MOVZWL R2,R0 ; Starting relative bit (page) in PFNMAP
54 51 FF 8F 78 00AC 199 ASHL #-16,R2,R1 ; Count of bits (pages) to look for
0000'CF 05 19 00B1 201 BSBB BOO$ALLOC_PAGES ; Settle for half if can't find all
0000'CF 52 7D 00B3 202 BLSS 20$ ; Go get the pages
0000'CF 55 8ED0 00B5 203 MOVQ R2,W^FIL$GQ_CACHE ; Failed
0000'CF 05 00BD 204 20$: POPL R5 ; Success, record the values
0000'CF 05 00BE 205 RSB ; Restore a register
0000'CF 05 00BE 206 :
0000'CF 05 00BE 207 : BOO$CACHE_INIT - Full routine to both allocate and open the cache
0000'CF 05 00BE 208 :
0000'CF 05 00BE 209 BOO$CACHE_INIT::
0000'CF 05 00BE 210 BSBB BOO$CACHE_ALLOC ; Allocate the cache
0000'CF 05 00C0 211 ; Fall thru to finish
0000'CF 05 00C0 212 :
0000'CF 05 00C0 213 : BOO$CACHE_OPEN - Actually mount the device and fill the cache
0000'CF 05 00C0 214 :
0000'CF 05 00C0 215 : BOO$CACHE_OPEN::
52 0000'CF D0 00C0 216 MOVL W^FIL$GQ_CACHE,R2 ; Pick up size
52 22 13 00C5 217 BEQL 10$ ; Zero length implies none
52 5E 04 C2 00C7 218 SUBL #4,SP ; Location to store channel
52 50 5E D0 00CA 219 MOVL SP,R0 ; Address to store channel
7E 52 02 C3 00CD 220 SUBL3 #2,R2,-(SP) ; Blocks in directory LBN cache
7E 0004'CF 09 78 00D1 221 PUSHL S^#<<1024-FIL$C_SIZE>/FIL$C_DIR_SIZE>; No. of dir cache entries
7E 52 09 78 00D3 222 ASHL #9,W^FIL$GQ_CACHE+4,-(SP) ; Byte address from page number
7E 52 09 78 00D9 223 ASHL #9,R2,-(SP) ; Size of cache in bytes
7E 52 7E D4 00DD 224 CLRL -(SP) ; Null device name string descriptor
0000'CF 50 DD 00DF 225 PUSHL R0 ; Address to store channel
0000'CF 06 FB 00E1 226 CALLS #6,W^FIL$CACHE_INIT ; Init the FIL$OPENFILE cache
0000'CF 5E 04 C0 00E6 227 ; descriptor returned in FIL$GQ_CACHE
0000'CF 05 00E9 228 ADDL #4,SP ; Clean off channel
0000'CF 05 00EA 229 10$: RSB
0000'CF 05 00EA 230 :
0000'CF 05 00EA 231 : BOO$ALLOC_PAGES - Find a run of contiguous, good pages
0000'CF 05 00EA 232 :
0000'CF 05 00EA 233 : Inputs:
0000'CF 05 00EA 234 R0 - Page to start at
0000'CF 05 00EA 235 R1 - Number of pages needed
0000'CF 05 00EA 236 R4 - Number willing to settle for
0000'CF 05 00EA 237 :
0000'CF 05 00EA 238 :
0000'CF 05 00EA 239 :

```

```

00EA 240 : R5 - Maximum page
00EA 241 : Outputs:
00EA 242 : CC - Status (BLSS to an error routine)
00EA 243 : R2 - Number found
00EA 244 : R3 - Starting page number
00EA 245 :
00EA 246 BOOS$ALLOC PAGES::
52 5B 17 9C 00EA 247 ROTL #<32-9>,R11,R2 ; PFN of the RPB
50 52 C0 00EE 248 ADDL R2,R0 ; Convert relative PFN to absolute
53 50 D0 00F1 249 MOVL R0,R3 ; Make a copy of starting bit
55 52 C0 00F4 250 ADDL R2,R5 ; Convert max relative PFN to absolute
50 55 D1 00F7 251 30$: CMPL R5,R0 ; Less than max page
1E 19 00FA 252 BLSS 50$ ; No, failure
OD 48 BB 50 E0 00FC 253 BBS R0,ARPBSQ_PFNMAP+4(R11),40$ ; Branch if this is a good page
52 50 53 C3 0101 254 SUBL3 R3,R0,R2 ; Count of bits (pages) found
54 52 D1 0105 255 CMPL R2,R4 ; Found enough?
10 18 0108 256 BGEQ 50$ ; Branch if yes
53 50 01 010A 257 ADDL3 #1,R0,R3 ; No, reset starting base
50 50 D6 010E 258 40$: INCL R0 ; Next bit (page)
E4 51 F5 0110 259 SOBGTR R1,30$ ; Branch if more to check
52 50 53 C3 0113 260 SUBL3 R3,R0,R2 ; Count of bits (pages) found
54 52 D1 0117 261 CMPL R2,R4 ; Found enough?
05 011A 262 50$: RSB ; Return (Status in CC)

```



```

011B 264 .SBTTL BOOSIMAGE_ATT - Get image attributes from image header
011B 265 :++
011B 266 : Functional Description:
011B 267 :
011B 268 : BOOSIMAGE_ATT returns to the caller some attributes of the image
011B 269 :
011B 270 : Calling Sequence:
011B 271 :
011B 272 : BSBW BOOSIMAGE_ATT
011B 273 :
011B 274 : Inputs:
011B 275 :
011B 276 : R2 = Size of file in blocks
011B 277 : R3 = Address of image header block (first one only)
011B 278 :
011B 279 : Outputs:
011B 280 :
011B 281 : R1 = Number of image header blocks at the front of the image
011B 282 : R2 = Size of image in blocks excluding the blocks at the end
011B 283 : containing local symbols, global symbols, or patch text
011B 284 :
011B 285 :--
011B 286 :
011B 287 BOOSIMAGE_ATT::
50 04 A3 3C 011B 288 MOVZWL IHD$W_SYMDBGOFF(R3),R0 ; ANY SYMBOL TABLE INFORMATION?
011B 289 BEQL 20$ ; BRANCH IF NOT
51 6043 9E 0121 290 MOVAB IHS$L_DSTVBN(R0)[R3],R1 ; ADR OF 1ST VBN IN DEBUG SYMBOL TABLE
011B 291 BSBB 40$ ; PROCESS IT
51 04 A043 9E 0127 292 MOVAB IHS$L_GSTVBN(R0)[R3],R1 ; ADR OF 1ST VBN IN GLOBAL SYMBOL TABLE
011B 293 BSBB 40$ ; PROCESS IT
50 08 A3 3C 012E 294 20$: MOVZWL IHD$W_PATCHOFF(R3),R0 ; ANY PATCH CONTROL INFORMATION?
011B 295 BEQL 30$ ; BRANCH IF NOT
51 20 A043 9E 0134 296 MOVAB IHP$L_PATCOMTXT(R0)[R3],R1 ; ADR OF 1ST VBN OF PATCH COMMAND TEXT
011B 297 BSBB 40$ ; PROCESS IT
51 10 A3 9A 013B 298 30$: MOVZBL IHD$B_HDRBLKCNT(R3),R1 ; GET IMAGE HEADER BLOCK COUNT
011B 299 RSB
0140 300 :
0140 301 : SEE IF VBN IS NON ZERO AND THEN IF IT IS SMALLER THAN THE CURRENT SMALLEST
0140 302 :
51 61 01 C3 0140 303 40$: SUBL3 #1,(R1),R1 ; FETCH VBN - 1
011B 304 BLSS 50$ ; BRANCH IF NO VBN IS PRESENT
51 52 D1 0146 305 CMPL R2,R1 ; IS IT SMALLER THAN THE CURRENT ONE
011B 306 BLEQ 50$ ; BRANCH IF NOT
52 51 D0 014B 307 MOVL R1,R2 ; YES, USE IT
011B 308 50$: RSB

```

```

014F 310      .SBTTL SYSS$ASSIGN, Dummy assign device system service
014F 311
014F 312      :++
014F 313      :
014F 314      : Functional description:
014F 315      :
014F 316      :     SYSS$ASSIGN is a dummy routine to satisfy the requirements of
014F 317      :     FIL$OPENFILE.
014F 318      :
014F 319      : Inputs:
014F 320      :
014F 321      :     CHAN(AP)      - address at which to return channel
014F 322      :
014F 323      : Implicit inputs:
014F 324      :
014F 325      :     The label BOO$GL_RPBBASE contains the physical address of the RPB.
014F 326      :
014F 327      : Outputs:
014F 328      :
014F 329      :     R0      - success status code
014F 330      :
014F 331      : Implicit outputs:
014F 332      :
014F 333      :     The channel returned is not a channel. It is instead the base
014F 334      :     address of the RPB.
014F 335      :
014F 336      :--
014F 337
00000008 014F 338 CHAN      = 8
014F 339
014F 340 SYSS$ASSIGN::      : Dummy system service.
0000      014F 341      .WORD      0
0151 342
08 BC 0000'CF D0 0151 343      MOVL      W^BOO$GL_RPBBASE,@CHAN(AP) ; Store RPB address as channel.
50 00' D0 0157 344      MOVL      S^#SS$_NORMAL,R0 ; Return success status.
04 015A 345      RET ; Return to caller.

```



```

015B 347 .SBTTL Common Globals for VMB and SYSBOOT
015B 348 :
015B 349 : The following globals are common to VMB and SYSBOOT and are
015B 350 : defined here to avoid replicate definitions.
015B 351 :
015B 352 FIL$GT_DDSTRING:: ; Default directory string.
5D 45 58 45 53 59 53 5B 00' 015B 353 .ASCIC /[SYSEXEC]/
08 015B
00 0164 354 FIL$GT_DDDEV:: ; Default device name
0164 355 .BYTE 0 ; Null ASCIC string
0165 356
0165 357 .END

```

BOOTIO
Symbol table

- BOOTSTRAP FILEREAD IO MODULE

J 6

15-SEP-1984 23:41:42 VAX/VMS Macro V04-00
4-SEP-1984 23:02:54 [BOOTS.SRC]BOOTIO.MAR;1

Page 11
(1)

BOOSALLOC_PAGES	000000EA	RG	02	OPS_CVTDH	= 000032FD
BOOSCACHE_ALLOC	0000008C	RG	02	OPS_CVTDL	= 0000006A
BOOSCACHE_INIT	000000BE	RG	02	OPS_CVTDW	= 00000069
BOOSCACHE_OPEN	000000C0	RG	02	OPS_CVTFB	= 00000048
BOOSGL_RPBASE	*****	X	02	OPS_CVTFD	= 00000056
BOOSIMAGE_ATT	0000011B	RG	02	OPS_CVTFG	= 000099FD
BUFADR	= 0000000C			OPS_CVTFH	= 000098FD
BYTCNT	= 00000014			OPS_CVTFL	= 0000004A
CHAN	= 00000008			OPS_CVTFW	= 00000049
FILSCACHE_INIT	*****	X	02	OPS_CVTGB	= 000048FD
FILSC_DIR_SIZE	*****	X	02	OPS_CVTGF	= 000033FD
FILSC_SIZE	*****	X	02	OPS_CVTGH	= 000056FD
FILSGD_CACHE	*****	X	02	OPS_CVTGL	= 00004AFD
FILSGT_DDDEV	00000164	RG	02	OPS_CVTGW	= 000049FD
FILSGT_DDSTRING	0000015B	RG	02	OPS_CVTHB	= 000068FD
FILSRDQRTLBN	00000000	RG	02	OPS_CVTHD	= 0000F7FD
INDSB_HDRBLKCNT	= 00000010			OPS_CVTHF	= 0000F6FD
INDSW_PATCHOFF	= 00000008			OPS_CVTHG	= 000076FD
INDSW_SYMDBGOFF	= 00000004			OPS_CVTHL	= 00006AFD
INPSL_PATCOMTXT	= 00000020			OPS_CVTHW	= 000069FD
INSSL_DSTVBN	= 00000000			OPS_CVTLD	= 0000006E
INSSL_GSTVBN	= 00000004			OPS_CVTLF	= 0000004E
IOFUNC	= 00000010			OPS_CVTLG	= 00004EFD
LBN	= 00000008			OPS_CVTLM	= 00006EFD
MEM_CACHE_TABLE	00000020	R	02	OPS_CVTLP	= 000000F9
OPS_ACB	= 0000006F			OPS_CVTPL	= 00000036
OPS_ACBF	= 0000004F			OPS_CVTPT	= 00000008
OPS_ACBG	= 00004FFD			OPS_CVTPT	= 00000024
OPS_ACBH	= 00006FFD			OPS_CVTRDL	= 0000006B
OPS_ADDD2	= 00000060			OPS_CVTRFL	= 0000004B
OPS_ADDD3	= 00000061			OPS_CVTRGL	= 00004BFD
OPS_ADDF2	= 00000040			OPS_CVTRHL	= 00006BFD
OPS_ADDF3	= 00000041			OPS_CVTSP	= 00000009
OPS_ADDG2	= 000040FD			OPS_CVTTP	= 00000026
OPS_ADDG3	= 000041FD			OPS_CVTWD	= 0000006D
OPS_ADDH2	= 000060FD			OPS_CVTWF	= 0000004D
OPS_ADDH3	= 000061FD			OPS_CVTWG	= 00004DFD
OPS_ADDP4	= 00000020			OPS_CVTWH	= 00006DFD
OPS_ADDP6	= 00000021			OPS_DIVD2	= 00000066
OPS_ASHP	= 000000F8			OPS_DIVD3	= 00000067
OPS_CLRD	= 0000007C			OPS_DIVF2	= 00000046
OPS_CLRF	= 000000D4			OPS_DIVF3	= 00000047
OPS_CLRG	= 0000007C			OPS_DIVG2	= 000046FD
OPS_CLRH	= 00007CFD			OPS_DIVG3	= 000047FD
OPS_CMPD	= 00000071			OPS_DIVH2	= 000066FD
OPS_CMPF	= 00000051			OPS_DIVH3	= 000067FD
OPS_CMPG	= 000051FD			OPS_DIVP	= 00000027
OPS_CMPP3	= 000071FD			OPS_EDITPC	= 00000038
OPS_CMPP4	= 00000035			OPS_EMODD	= 00000074
OPS_CRC	= 00000037			OPS_EMODF	= 00000054
OPS_CVTBD	= 0000000B			OPS_EMODG	= 000054FD
OPS_CVTBF	= 0000006C			OPS_EMODH	= 000074FD
OPS_CVTBG	= 0000004C			OPS_MATCHC	= 00000039
OPS_CVTBH	= 00004CFD			OPS_MNEGD	= 00000072
OPS_CVTDB	= 00006CFD			OPS_MNEGF	= 00000052
OPS_CVTDF	= 00000068			OPS_MNEGG	= 000052FD
	= 00000076			OPS_MNEGH	= 000072FD

BOOTIO
Symbol table

- BOOTSTRAP FILEREAD IO MODULE

K 6

15-SEP-1984 23:41:42 VAX/VMS Macro V04-00
4-SEP-1984 23:02:54 [BOOTS.SRC]BOOTIO.MAR;1

Page 12
(1)

```

OPS_MOVD      = 00000070
OPS_MOVF      = 00000050
OPS_MOVG      = 000050FD
OPS_MOVH      = 000070FD
OPS_MOVP      = 00000034
OPS_MOVTC     = 0000002E
OPS_MOVTUC    = 0000002F
OPS_MULD2     = 00000064
OPS_MULD3     = 00000065
OPS_MULF2     = 00000044
OPS_MULF3     = 00000045
OPS_MULG2     = 000044FD
OPS_MULG3     = 000045FD
OPS_MULH2     = 000064FD
OPS_MULH3     = 000065FD
OPS_MULP      = 00000025
OPS_POLYD     = 00000075
OPS_POLYF     = 00000055
OPS_POLYG     = 000055FD
OPS_POLYH     = 000075FD
OPS_SCANC     = 0000002A
OPS_SKPC      = 0000003B
OPS_SPANC     = 0000002B
OPS_SUBD2     = 00000062
OPS_SUBD3     = 00000063
OPS_SUBF2     = 00000042
OPS_SUBF3     = 00000043
OPS_SUBG2     = 000042FD
OPS_SUBG3     = 000043FD
OPS_SUBH2     = 000062FD
OPS_SUBH3     = 000063FD
OPS_SUBP4     = 00000022
OPS_SUBP6     = 00000023
OPS_TSTD      = 00000073
OPS_TSTF      = 00000053
OPS_TSTG      = 000053FD
OPS_TSTH      = 000073FD
RPBSL_IOVEC   = 00000034
RPBSL_PFNENT  = 0000004C
RPBSQ_PFNMAP  = 00000044
SS$ NORMAL    ***** X 02
SYS$ASSIGN    0000014F RG 02

```

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
YFILEREAD	00000165 (357.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Initialization	30	00:00:00.10	00:00:00.31
Command processing	110	00:00:00.72	00:00:01.86
Pass 1	406	00:00:11.31	00:00:20.88
Symbol table sort	0	00:00:00.79	00:00:01.02
Pass 2	76	00:00:03.69	00:00:07.01
Symbol table output	19	00:00:00.15	00:00:00.20
Psect synopsis output	1	00:00:00.03	00:00:00.34
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	644	00:00:16.80	00:00:31.63

The working set limit was 1500 pages.
49198 bytes (97 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 559 non-local and 10 local symbols.
3109 source lines were read in Pass 1, producing 14 object records in Pass 2.
139 pages of virtual memory were used to define 138 macros.

! Macro library statistics !

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[BOOTS.OBJ]BOOTS.MLB;1	0
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	4
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	8

613 GETS were required to define 8 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:BOOTIO/OBJ=OBJ\$:BOOTIO MASD\$:[EMULAT.SRC]MISSING/UPDATE=(MASD\$:[EMULAT.ENH]MISSING)+MASD\$:[BOOTS.SRC]BOOTIO/UPDATE=(M

0037 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

